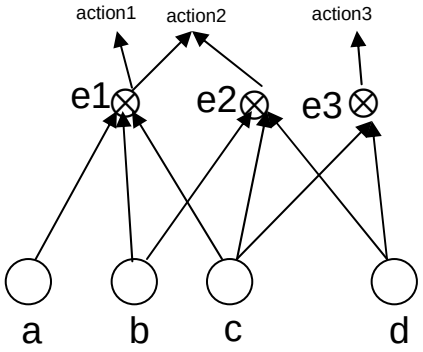
Data-Driven Backward Chaining Rules for CEP

Darko Anicic, Paul Fodor, Roland Stühmer, Nenad Stojanovic

Sequential



Original program:

a x b x c -> e1

b x c x d -> e2

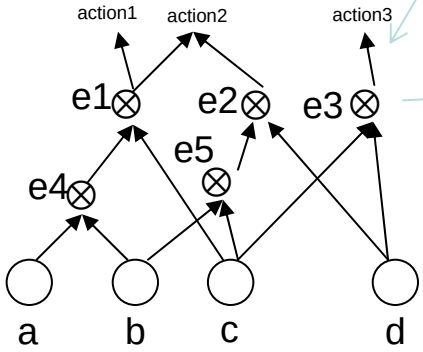
c x d -> e3

Action triggers:

e1 -> action1.

e1 or e2 -> action2.

e3 -> action3.

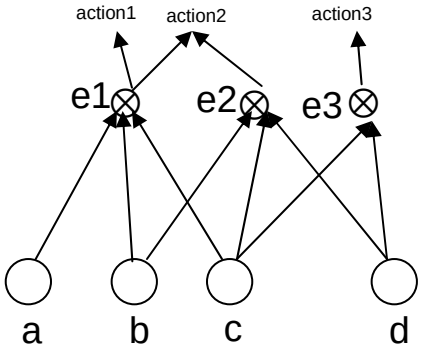



```
a :- while_do(a,1).
a(1) :- ins(goal(b,a,e4)).
a(1) :- true.

b :- while_do(b,1).
b(1) :- goal(b,a,e4) * del(goal(b,a,e4)) * e4.
b(1) :- true.
b(2) :- ins(goal(c,b,e5)).
b(2) :- true.

c :- while_do(c,1).
c(1) :- goal(c,e4,e1) * del(goal(c,e4,e1)) * e1.
c(1) :- true.
c(2) :- goal(c,b,e5) * del(goal(c,b,e5)) * e5.
c(2) :- true.
c(3) :- ins(goal(d,c,e3)).
c(3) :- true.

d :- while_do(d,1).
d(1) :- goal(d,e5,e2) * del(goal(d,e5,e2)) * e2.
d(1) :- true.
d(2) :- goal(d,c,e3) * del(goal(d,c,e3)) * e3.
d(2) :- true.

while_do(Pred,N):-
        (FullPred =.. [Pred,N]) *
        execCTR(FullPred) *
        (N1 is N+1) *
        while_do(Pred,N1).
while_do(Pred,N):- true.
```

```
e1 :- while_do(e1,1).
e1(1) :- action1.
e1(1) :- true.
e1(2) :- action2.
e1(2) :- true.

e2 :- while_do(e2,1).
e2(1) :- action2.
e2(1) :- true.

e3 :- while_do(e1,1).
e3(1) :- action3.
e3(1) :- true.

e4 :- while_do(e4,1).
e4(1) :- ins(goal(c,e4,e1)).
e4(1) :- true.

e5 :- while_do(e5,1).
e5(1) :- ins(goal(d,e5,e2)).
e5(1) :- true.


action1:- write('action1'),nl.
action2:- write('action2'),nl.
action3:- write('action3'),nl.
```

```
xsb -e "[load],init,ctr_comp('event_01')."

 | ?- execCTR(a).
 yes
 | ?- execCTR(b).
 yes
 | ?- execCTR(c).
 action1
 action2
 yes
 | ?- execCTR(d).
action3
 yes
```

Sequential with times



Original program:

a([T1,T2]) * b([T3,T4]) * c([T5,T6]) -> e1([T1,T6])

b([T1,T2]) * c([T3,T4]) * d([T5,T6]) -> e2([T1,T6])

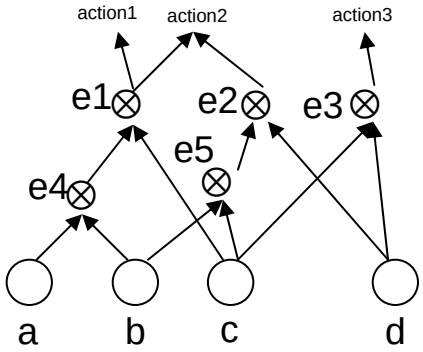
c([T1,T2]) * d([T3,T4]) -> e3([T1,T4])

Action triggers:

e1([T1,T2]) -> action1([T1,T2]).

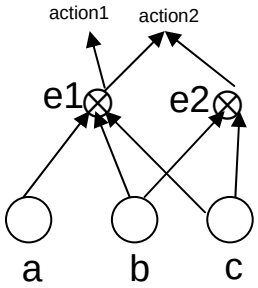
e1([T1,T2]) or e2([T1,T2]) -> action2([T1,T2]).

e3([T1,T2]) -> action3([T1,T2]).




```
a([T1,T2]) :- while_do(a,1,[T1,T2]).
a(1,[T1,T2]) :- ins(goal(b(_,_),a([T1,T2]),e4(_,_))).

b([T1,T2]) :- while_do(b,1,[T1,T2]).
b(1,[T3,T4]) :- goal(b([T3,T4]),a([T1,T2]),e4(_,_)) * T2<T3 * del(goal(b([T3,T4]),a([T1,T2]),e4(_,_))) * e4([T1,T4]).
b(1,[T3,T4]) :- true.
b(2,[T1,T2]) :- ins(goal(c(_,_),b([T1,T2]),e5(_,_))).

c([T1,T2]) :- while_do(c,1,[T1,T2]).
c(1,[T3,T4]) :- goal(c([T3,T4]),e4([T1,T2]),e1(_,_)) * T2<T3 * del(goal(c([T3,T4]),e4([T1,T2]),e1(_,_))) *
e1([T1,T4]).
c(1,[T3,T4]) :- true.
c(2,[T3,T4]) :- goal(c([T3,T4]),b([T1,T2]),e5(_,_)) * T2<T3 * del(goal(c([T3,T4]),b([T1,T2]),e5(_,_))) * e5([T1,T4]).
c(2,[T3,T4]) :- true.
c(3,[T1,T2]) :- ins(goal(d(_,_),c([T1,T2]),e3(_,_))).

d([T1,T2]) :- while_do(d,1,[T1,T2]).
d(1,[T3,T4]) :- goal(c([T3,T4]),e5([T1,T2]),e2(_,_)) * T2<T3 * del(goal(c([T3,T4]),e5([T1,T2]),e2(_,_))) *
e2([T1,T4]).
d(1,[T3,T4]) :- true.
d(2,[T3,T4]) :- goal(d([T3,T4]),c([T1,T2]),e3(_,_)) * T2<T3 * del(goal(d([T3,T4]),c([T1,T2]),e3(_,_))) * e3([T1,T4]).

e1([T1,T2]) :- while_do(e1,1,[T1,T2]).
e1(1,[T1,T2]) :- action1([T1,T2]).
e1(1,[T1,T2]) :- true.
e1(2,[T1,T2]) :- action2([T1,T2]).

e2([T1,T2]) :- while_do(e2,1,[T1,T2]).
e2(1,[T1,T2]) :- action2([T1,T2]).
e2(1,[T1,T2]) :- true.

e3([T1,T2]) :- while_do(e3,1,[T1,T2]).
e3(1,[T1,T2]) :- action3([T1,T2]).

e4([T1,T2]) :- while_do(e4,1,[T1,T2]).
e4(1,[T1,T2]) :- ins(goal(c(_,_),e4([T1,T2]),e1(_,_))).

e5([T1,T2]) :- while_do(e5,1,[T1,T2]).
e5(1,[T1,T2]) :- ins(goal(c(_,_),e5([T1,T2]),e2(_,_))).

action1([T1,T2]):- write(action1([T1,T2])),nl.
action2([T1,T2]):- write(action2([T1,T2])),nl.
action3([T1,T2]):- write(action3([T1,T2])),nl.

while_do(Pred,N,L):- (FullPred =.. [Pred,N,L]) * execCTR(FullPred) * (N1 is N+1) * while_do(Pred,N1,L).
while_do(Pred,N,L):- true.
```

```
xsb -e "[load],init,ctr_comp('event_01_withTimes')."

 | ?- execCTR(a([1,1])).
 yes
 | ?- execCTR(b([2,2])).
 yes
 | ?- execCTR(c([3,3])).
 action1([1,3])
 action2 ([1,3])
 yes
 | ?- execCTR(d ([4,4])).
action3([3,4])
 yes
```



Original program:

a([T1,T2]) * b([T3,T4]) * c([T5,T6]) -> e1([T1,T6])

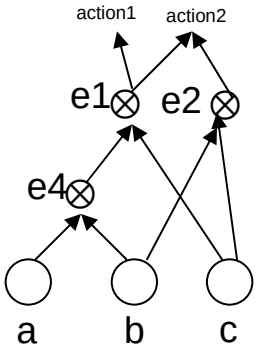
b([T1,T2]) * c([T3,T4]) -> e2([T1,T6])

Action triggers:

e1([T1,T2]) -> action1([T1,T2]).

e1([T1,T2]) or e2([T1,T2]) -> action2([T1,T2]).

Out-of-order sequence with times



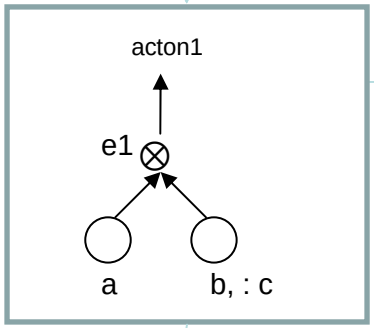

```
a([T1,T2]) :- while_do(a,1,[T1,T2]).
a(1,[T1,T2]) :- ins(goal(b([_,_]),a([T1,T2]),e4([_,_]))).
a(2,[T3,T4]) :- goal_o(a([_,_]),b([T1,T2]),e4([_,_])) * T4<T1 * del(goal_o(a([_,_]),b([T1,T2]),e4([_,_]))) *
e4([T3,T2]).

b([T1,T2]) :- while_do(b,1,[T1,T2]).
b(1,[T3,T4]) :- goal(b([_,_]),a([T1,T2]),e4([_,_])) * T2<T3 * del(goal(b([_,_]),a([T1,T2]),e4([_,_]))) * e4([T1,T4]).
b(2,[T1,T2]) :- ins(goal_o(a([_,_]),b([T1,T2]),e4([_,_]))). // DELTE using K-Clack!
b(3,[T1,T2]) :- ins(goal(c([_,_]),b([T1,T2]),e5([_,_]))).
b(4,[T3,T4]) :- goal(b([_,_]),c([T1,T2]),e5([_,_])) * T4<T1 * del(goal(b([_,_]),c([T1,T2]),e5([_,_]))) * e5([T3,T2]).

c([T1,T2]) :- while_do(c,1,[T1,T2]).
c(1,[T3,T4]) :- goal(c([_,_]),e4([T1,T2]),e1([_,_])) * T2<T3 * del(goal(c([_,_]),e4([T1,T2]),e1([_,_]))) * e1([T1,T4]).
c(2,[T1,T2]) :- ins(goal_o(e4([_,_]),c([T1,T2]),e1([_,_]))).
c(3,[T3,T4]) :- goal(c([_,_]),b([T1,T2]),e2([_,_])) * T2<T3 * del(goal(c([_,_]),b([T1,T2]),e2([_,_]))) * e2([T1,T4]).
c(4,[T1,T2]) :- ins(goal_o(b([_,_]),c([T1,T2]),e5([_,_]))).

e4([T1,T2]) :- while_do(e4,1,[T1,T2]).
e4(1,[T1,T2]) :- ins(goal(c([_,_]),e4([T1,T2]),e1([_,_]))).
e4(2,[T3,T4]) :- goal_o(e4([_,_]),c([T1,T2]),e1([_,_])) * T4<T1 * del(goal_o(e4([_,_]),c([T1,T2]),e1([_,_]))) *
e1([T3,T2]).

e1([T1,T2]) :- while_do(e1,1,[T1,T2]).
e1(1,[T1,T2]) :- action1([T1,T2]).
e1(2,[T1,T2]) :- action2([T1,T2]).

e2([T1,T2]) :- while_do(e2,1,[T1,T2]).
e2(1,[T1,T2]) :- action2([T1,T2]).

while_do(Pred,N,L):- (FullPred =.. [Pred,N,L]) * execCTR(FullPred) * (N1 is N+1) * while_do(Pred,N1,L).
while_do(Pred,N,L):- true.
```

```
xsb -e
"[load],init,ctr_comp('event_01_withTimes')."

| ?- execCTR(a([1,1])).
yes
| ?- execCTR(b([2,2])).
yes
| ?- execCTR(c([3,3])).
action1([1,3])
action2 ([1,3])
yes
```

Original program: (a x b) /\ not c -> e1

Informally, event a was fired followed by event b, where event c was not encounted on the execution path.

(a([T1,T2]) * b([T3,T4])) /\ not c([T5,T6]) -> e1([T1,T4])



Out-of-order negation with times

```
a([T1,T2]) :- while_do(a,1,[T1,T2]).
a(1,[T1,T2]) :- ins(goal(b([_,_]),a([T1,T2]),e1([_,_]))).
a(2,[T5,T6]) :- goal_o(a([_,_]),b([T1,T2]),e1([_,_])) * not(goal(_,c([T3,T4]),_)) * T6<T1 *
    * T5<T3 * T4<T2 * del(goal_o(a([_,_]),b([T1,T2]),e1([_,_]))) * e1([T5,T2]).

b([T1,T2]) :- while_do(b,1,[T1,T2]).
b(1,[T5,T6]) :- goal(b([_,_]),a([T1,T2]),e1([_,_])) * not(goal(_,c([T3,T4]),_)) * T2<T5 *
T1<T3 * T4<T6 * del(goal(b([T3,T4]),a([T1,T2]),e1([_,_]))) * e1([T1,T4]).
b(2,[T1,T2]) :- ins(goal_o(a([_,_]),b([T1,T2]),e1([_,_]))). // DELTE using K-Clack!

c([T1,T2]) :- while_do(c,1,[T1,T2]).
c(1,[T1,T2]) :- ins(goal(_,c([T1,T2]),_)).
c(2,[T3,T4]) :- e1([T1,T2]) * T1<T3=<T4<T2 * retract(e1 ([T1,T2])).//out-of-order event

e1([T1,T2]) :- while_do(e1,1,[T1,T2]).
e1(1,[T1,T2]) :- action1.

retract(e1 ([T1,T2])) :- write(send another event re1([T1,T2])),nl.

action1([T1,T2]):- write(action1([T1,T2])),nl.

while_do(Pred,N,L):- (FullPred =.. [Pred,N,L]) * execCTR(FullPred) * (N1 is N+1) *
while_do(Pred,N1,L).
while_do(Pred,N,L):- true.
```

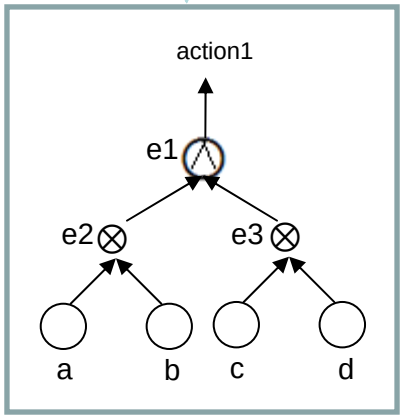
```
xsb -e "[load],init,ctr_comp('event_03_withTimes')."

?- execCTR(a([1,1])),execCTR(b([2,2])).
 action1([1,2])

?- execCTR(a([1,1])),execCTR(c([2,2])),execCTR(b([3,3])).
no action was triggered
```

Original program:

(a x b) /\ (c x d) -> e1

Action triggers:

e1 -> action1.




```
a :- while_do(a,1).
a(1) :- ins(goal(b,a,e2)).

b :- while_do(b,1).
b(1) :- goal(b,a,e2) * del(goal(b,a,e2)) * e2.

c :- while_do(c,1).
c(1) :- ins(goal(d,c,e3)).

d :- while_do(d,1).
d(1) :- goal(d,c,e3) * del(goal(d,c,e3)) * e3.

e1 :- while_do(e1,1).
e1(1) :- action1.

e2 :- while_do(e2,1).
e2(1) :- goal(e2,e3,e1) * del(goal(e2,e3,e1)) * e1.
e2(1) :- not(goal(e2,e3,e1)) * ins(goal(e3,e2,e1)).

e3 :- while_do(e3,1).
e3(1) :- goal(e3,e2,e1) * del(goal(e3,e2,e1)) * e1.
e3(1) :- not(goal(e3,e2,e1)) * ins(goal(e2,e3,e1)).

action1:- write('action1'),nl.

while_do(Pred,N):- (FullPred =.. [Pred,N]) * execCTR(FullPred) * (N1 is N+1) * while_do(Pred,N1).
while_do(Pred,N):- true.
```

Classical (concurrent) conjunction

```
xsb -e "[load],init,ctr_comp('event_02')."

 | ?- execCTR(a),execCTR(b),execCTR(c),execCTR(d).
 action1

| ?- execCTR(a),execCTR(c),execCTR(b),execCTR(d).
 action1

| ?- execCTR(a),execCTR(c),execCTR(d),execCTR(b).
 action1

| ?- execCTR(c),execCTR(a),execCTR(b),execCTR(d).
 action1

| ?- execCTR(c),execCTR(a),execCTR(d),execCTR(b).
 action1

| ?- execCTR(c),execCTR(d),execCTR(a),execCTR(b).
 action1
```

Original program:

a([T1,T2]) * b([T3,T4]) -> e2([T1,T4])

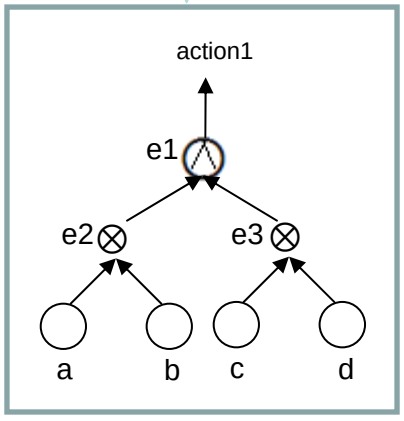
c([T1,T2]) * d([T3,T4]) -> e3([T1,T4])

e2([T1,T2]) /\ e3([T3,T4]) -> e1([T5,T6])

Note: Unlike in a sequential composition, here the order of occurrence of d and c is not important. The only important thing is that T5=min(T1,T3) and T6=max(T2,T4).

Action triggers:

e1([T1,T2]) -> action1([T1,T2]).




```
a([T1,T2]) :- while_do(a,1,[T1,T2]).
a(1,[T1,T2]) :- ins(goal(b([_,_]),a([T1,T2]),e2([_,_]))).

b([T1,T2]) :- while_do(b,1,[T1,T2]).
b(1,[T3,T4]) :- goal(b([T3,T4]),a([T1,T2]),e2([_,_])) * del(goal(b([T3,T4]),a([T1,T2]),e2([_,_]))) * e2([T1,T4]).

c([T1,T2]) :- while_do(c,1,[T1,T2]).
c(1,[T1,T2]) :- ins(goal(d([_,_]),c([T1,T2]),e3([_,_]))).

d([T1,T2]) :- while_do(d,1,[T1,T2]).
d(1,[T3,T4]) :- goal(d([T3,T4]),c([T1,T2]),e3([_,_])) * del(goal(d([T3,T4]),c([T1,T2]),e3([_,_]))) * e3([T1,T4]).

e1([T1,T2]) :- while_do(e1,1,[T1,T2]).
e1(1,[T1,T2]) :- action1.

e2([T1,T2]) :- while_do(e2,1,[T1,T2]).
e2(1,[T3,T4]) :- goal(e2([T3,T4]),e3([T1,T2]),e1([_,_])) * del(goal(e2([T3,T4]),e3([T1,T2]),e1([_,_]))) * min(T1,T3,T5) * max(T2,T4,T6)  *
e1([T5,T6]).
e2(1,[T3,T4]) :- not(goal(e2([T3,T4]),e3([_,_]),e1([_,_]))) * ins(goal(e3([_,_]),e2([T3,T4]),e1([_,_]))).

e3([T1,T2]) :- while_do(e3,1,[T1,T2]).
e3(1,[T3,T4]) :- goal(e3([T3,T4]),e2([T1,T2]),e1([_,_])) * del(goal(e3([T3,T4]),e2([T1,T2]),e1([_,_]))) * min(T1,T3,T5) * max(T2,T4,T6) *
e1([T5,T6]).
e3(1,[T3,T4]) :- not(goal(e3([T3,T4]),e2([_,_]),e1([_,_]))) * ins(goal(e2([_,_]),e3([T3,T4]),e1([_,_]))).

action1:- write('action1'),nl.

while_do(Pred,N,L):- (FullPred =.. [Pred,N,L]) * execCTR(FullPred) * (N1 is N+1) * while_do(Pred,N1,L).
while_do(Pred,N,L):- true.

min(T1,T2,T3):- ( T1 < T2 -> T3 = T1 ; T3 = T2 ).
```

Classical (concurrent) conjunction with times

```
xsb -e "[load],init,ctr_comp('event_02_withTimes')."

 ?- execCTR(a([1,1])),execCTR(b([2,2])),execCTR(c([3,3])),execCTR(d([4,4])).
action1([1,4])

?- execCTR(a([1,1])),execCTR(c([2,2])),execCTR(b([3,3])),execCTR(d([4,4])).
action1 ([1,4])

?- execCTR(a([1,1])),execCTR(c([2,2])),execCTR(d([3,3])),execCTR(b([4,4])).
action1 ([1,4])

?- execCTR(c([1,1])),execCTR(a([2,2])),execCTR(b([3,3])),execCTR(d([4,4])).
action1 ([1,4])

?- execCTR(c([1,1])),execCTR(a([2,2])),execCTR(d([3,3])),execCTR(b([4,4])).
action1 ([1,4])

?- execCTR(c([1,1])),execCTR(d([2,2])),execCTR(a([3,3])),execCTR(b([4,4])).
action1 ([1,4])
```

Original program:

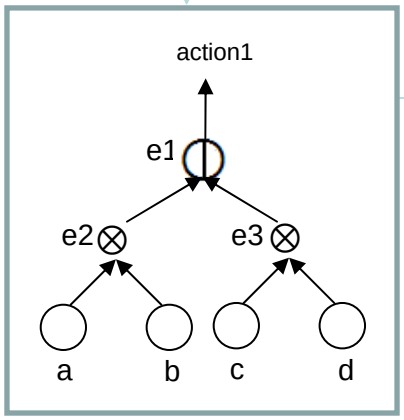
a([T1,T2]) * b([T3,T4]) -> e2([T1,T4])

c([T1,T2]) * d([T3,T4]) -> e3([T1,T4])

e2([T1,T2]) | e3([T1,T2]) -> e1([T1,T2])

Action triggers:

e1([T1,T2]) -> action1([T1,T2]).



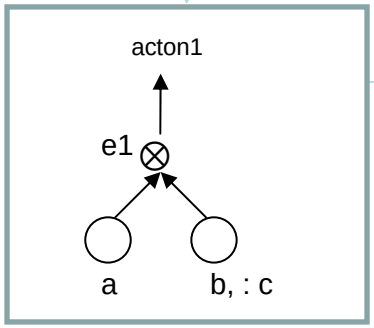

```
a([T1,T2]) :- while_do(a,1,[T1,T2]).
a(1,[T1,T2]) :- ins(goal(b([_,_]),a([T1,T2]),e2([_,_]))).

b([T1,T2]) :- while_do(b,1,[T1,T2]).
b(1,[T3,T4]) :- goal(b([T3,T4]),a([T1,T2]),e2([_,_])) * del(goal(b([T3,T4]),a([T1,T2]),e2([_,_]))) * e2([T1,T4]).

c([T1,T2]) :- while_do(c,1,[T1,T2]).
c(1,[T1,T2]) :- ins(goal(d([_,_]),c([T1,T2]),e3([_,_]))).

d([T1,T2]) :- while_do(d,1,[T1,T2]).
d(1,[T3,T4]) :- goal(d([T3,T4]),c([T1,T2]),e3([_,_])) * del(goal(d([T3,T4]),c([T1,T2]),e3([_,_]))) * e3([T1,T4]).

e1([T1,T2]) :- while_do(e1,1,[T1,T2]).
e1(1,[T1,T2]) :- action1.

e2([T1,T2]) :- while_do(e2,1,[T1,T2]).
e2(1,[T3,T4]) :- goal(e2([T3,T4]),e3([T1,T2]),e1([_,_])) * del(goal(e2([T3,T4]),e3([T1,T2]),e1([_,_]))) * T3<T2 * e1([T1,T4]).
e2(1,[T3,T4]) :- not(goal(e2([T3,T4]),e3([_,_]),e1([_,_]))) * ins(goal(e3([_,_]),e2([T3,T4]),e1([_,_]))).

e3([T1,T2]) :- while_do(e3,1,[T1,T2]).
e3(1,[T3,T4]) :- goal(e3([T3,T4]),e2([T1,T2]),e1([_,_])) * del(goal(e3([T3,T4]),e2([T1,T2]),e1([_,_]))) * T3<T2 * e1([T1,T4]).
e3(1,[T3,T4]) :- not(goal(e3([T3,T4]),e2([_,_]),e1([_,_]))) * ins(goal(e2([_,_]),e3([T3,T4]),e1([_,_]))).

action1([T1,T2]):- write(action1([T1,T2])),nl.

while_do(Pred,N,L):- (FullPred =.. [Pred,N,L]) * execCTR(FullPred) * (N1 is N+1) * while_do(Pred,N1,L).
while_do(Pred,N,L):- true.
```

Classical (concurrent) conjunction with time overlapping

```
xsb -e "[load],init,ctr_comp('event_02_withTimesOverlapping')."

 ?- execCTR(a([1,1])),execCTR(b([2,2])),execCTR(c([3,3])),execCTR(d([4,4])).
no action triggered

?- execCTR(a([1,1])),execCTR(c([2,2])),execCTR(b([3,3])),execCTR(d([4,4])).
action1 ([1,4])

?- execCTR(a([1,1])),execCTR(c([2,2])),execCTR(d([3,3])),execCTR(b([4,4])).
action1 ([1,4])

?- execCTR(c([1,1])),execCTR(a([2,2])),execCTR(b([3,3])),execCTR(d([4,4])).
action1 ([1,4])

?- execCTR(c([1,1])),execCTR(a([2,2])),execCTR(d([3,3])),execCTR(b([4,4])).
action1 ([1,4])

?- execCTR(c([1,1])),execCTR(d([2,2])),execCTR(a([3,3])),execCTR(b([4,4])).
no action triggered
```

Original program: (a x b) /\ not c -> e1

Informally, event a was fired followed by event b, where event c was not encounted on the execution path.



Negation

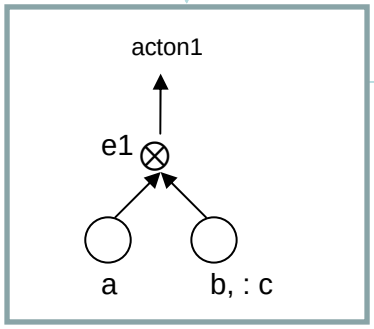
```
a :- while_do(a,1).
a(1) :- ins(goal(b,a,e1)).

b :- while_do(b,1).
b(1) :- goal(b,a,e1) * not(goal(_,c,_)) * del(goal(b,a,e1)) * e1.

c :- while_do(c,1).
c(1) :- ins(goal(_,c,_)).

e1 :- while_do(e1,1).
e1(1) :- action1.

action1:- write('action1'),nl.

while_do(Pred,N):- (FullPred =.. [Pred,N]) * execCTR(FullPred) * (N1 is N+1) * while_do(Pred,N1).
while_do(Pred,N):- true.
```

```
xsb -e "[load],init,ctr_comp('event_03‘)."

?- execCTR(a),execCTR(b).
 action1

?- execCTR(a),execCTR(c),execCTR(b).
no action was triggered
```

Original program: (a x b) /\ not c -> e1

Informally, event a was fired followed by event b, where event c was not encounted on the execution path.

(a([T1,T2]) * b([T3,T4])) /\ not c([T5,T6]) -> e1([T1,T4])



Negation with times

```
a([T1,T2]) :- while_do(a,1,[T1,T2]).
a(1,[T1,T2]) :- ins(goal(b([_,_]),a([T1,T2]),e1([_,_]))).

b([T1,T2]) :- while_do(b,1,[T1,T2]).
b(1,[T5,T6]) :- goal(b([_,_]),a([T1,T2]),e1([_,_])) * not(goal(_,c([T3,T4]),_)) * T2<T5 *
T1<T3 * T4<T6 * del(goal(b([T3,T4]),a([T1,T2]),e1([_,_]))) * e1([T1,T4]).

c([T1,T2]) :- while_do(c,1,[T1,T2]).
c(1,[T1,T2]) :- ins(goal(_,c([T1,T2]),_)).

e1([T1,T2]) :- while_do(e1,1,[T1,T2]).
e1(1,[T1,T2]) :- action1.

action1([T1,T2]):- write(action1([T1,T2])),nl.

while_do(Pred,N,L):- (FullPred =.. [Pred,N,L]) * execCTR(FullPred) * (N1 is N+1) *
while_do(Pred,N1,L).
while_do(Pred,N,L):- true.
```

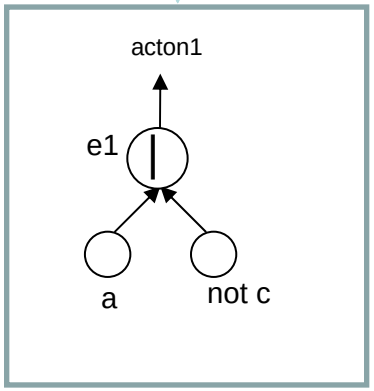
```
xsb -e "[load],init,ctr_comp('event_03_withTimes')."

?- execCTR(a([1,1])),execCTR(b([2,2])).
 action1([1,2])

?- execCTR(a([1,1])),execCTR(c([2,2])),execCTR(b([3,3])).
no action was triggered
```

Original program:

a |not c -> e1

Informally, event “a” was fired and event “c” was not encountered on the execution path of “a”.

a([T1,T2]) | not c([T3,T4]) -> e1([T1,T2])




```
a([T1,T2]) :- while_do(b,1,[T1,T2]).
a(1,[T1,T2]) :- {not(goal(_,c([T3,T4]),_)) \/ {goal(_,c([T3,T4]),_) * {[T4<T1] \/
[T2<T3]} } } * e1([T1,T2]).

c([T1,T2]) :- while_do(c,1,[T1,T2]).
c(1,[T1,T2]) :- ins(goal(_,c([T1,T2]),_)).

e1([T1,T2]) :- while_do(e1,1,[T1,T2]).
e1(1,[T1,T2]) :- action1.

action1([T1,T2]):- write(action1([T1,T2])),nl.

while_do(Pred,N,L):- (FullPred =.. [Pred,N,L]) * execCTR(FullPred) * (N1 is
N+1) * while_do(Pred,N1,L).
while_do(Pred,N,L):- true.
```

Negation with times and concur.conj. (binary events only)

?- execCTR(a([1,2])) and not c occures in [1,2].
action1([1,2])

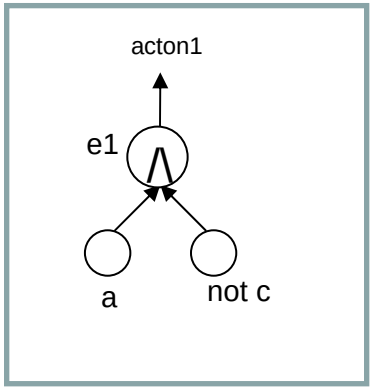
?- execCTR(a([1,4])),execCTR(b([2,3])).
no action was triggered

Original program:

a /\ not c -> e1([T1,T6])

Informally, an interval [T1,T6] is observed. Event “a” was fired on that interval and event “c” did not happen either before, after or during “a” (within that interval).

a([T1,T2]) /\ not c([T3,T4]) -> e1([T1,T6])




```
a([T1,T2]) :- while_do(b,1,[T1,T2]).
a(1,[T2,T3]) :- {not(goal(_,c([T4,T5]),_)) \/ {goal(_,c([T4,T5]),_) * {[T5<T1] \/ [T6<T4]} } } * e1([T1,T6]).

c([T1,T2]) :- while_do(c,1,[T1,T2]).
c(1,[T1,T2]) :- ins(goal(_,c([T1,T2]),_)).

e1([T1,T2]) :- while_do(e1,1,[T1,T2]).
e1(1,[T1,T2]) :- action1.

action1([T1,T2]):- write(action1([T1,T2])),nl.

while_do(Pred,N,L):- (FullPred =.. [Pred,N,L]) * execCTR(FullPred) * (N1 is N+1) * while_do(Pred,N1,L).
while_do(Pred,N,L):- true.
```

```
?- execCTR(a([1,2])).
 action1([1,2])

?- execCTR(a([1,4])),execCTR(b([2,3])).
no action was triggered
```

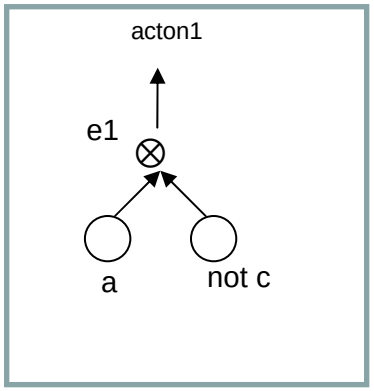
Negation with times and class.conj. (binary events only)

Original program:

a x not c -> e1([T1,T5])

Informally, an interval [T1,T5] is observed. Event “a” was fired on that interval and event “c” did not happen after “a” (within that interval).

a([T1,T2]) x not c([T3,T4]) -> e1([T1,T5])




```
a([T1,T2]) :- while_do(b,1,[T1,T2]).
a(1,[T1,T2]) :- {not(goal(_,c([T3,T4]),_)) \/ {goal(_,c([T3,T4]),_) * T5<T3}} *
e1([T1,T5]).

c([T1,T2]) :- while_do(c,1,[T1,T2]).
c(1,[T1,T2]) :- ins(goal(_,c([T1,T2]),_)).

e1([T1,T2]) :- while_do(e1,1,[T1,T2]).
e1(1,[T1,T2]) :- action1.

action1([T1,T2]):- write(action1([T1,T2])),nl.

while_do(Pred,N,L):- (FullPred =.. [Pred,N,L]) * execCTR(FullPred) * (N1 is
N+1) * while_do(Pred,N1,L).
while_do(Pred,N,L):- true.
```

```
?- execCTR(a([1,2])).
 action1([1,2])

?- execCTR(a([1,4])),execCTR(b([2,3])).
no action was triggered
```

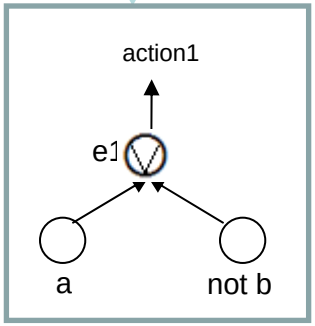
Negation with times and ser.conj. (binary events only)

Original program:

a ∨ not b -> e1([T1,T6])

Informally, an interval [T1,T4] is observed. E1is triggered when either “a” was fired on that interval or “b” did not happen within that interval.

a([T2,T3]) ∨ not b([T4,T5]) -> e1([T1,T6])




```
a([T2,T3]) :- while_do(a,1,[T2,T3]).
a(1,[T2,T3]) :- T1<T2 * T3<T4 * e1([T1,T6]).

b([T1,T2]) :- while_do(b,1,[T1,T2]).
b(1,[T1,T2]) :- ins(goal(_,b([T1,T2]),_)).

e1([T1,T2]) :- while_do(e1,1,[T1,T2]).
e1(1,[T1,T2]) :- action1.
e1(2,[T1,T6]) :- measure?[T1,T6] * {not b[T2,T3] ∨ [b[T2,T3] * (T3<T1 ∨
T6<T2)]} * action1.

action1([T1,T2]):- write(action1([T1,T2])),nl.

while_do(Pred,N,L):- (FullPred =.. [Pred,N,L]) * execCTR(FullPred) * (N1 is
N+1) * while_do(Pred,N1,L).
while_do(Pred,N,L):- true.
```

```
?- execCTR(a([1,2])).
 action1([1,2])

?- execCTR(a([1,4])),execCTR(b([2,3])).
no action was triggered
```

Negation with times and disjunction (binary events only)

Disjunction

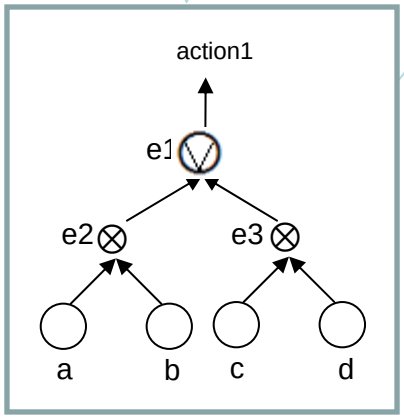
Original program:

(a x b) \/ (c x d) -> e1

Action triggers:

e1 -> action1.

Note: disjunction is just transformed into disjunctive Horn rules.




```
a :- while_do(a,1).
a(1) :- ins(goal(b,a,e2)).

b :- while_do(b,1).
b(1) :- goal(b,a,e2) * del(goal(b,a,e2)) * e2.

c :- while_do(c,1).
c(1) :- ins(goal(d,c,e3)).

d :- while_do(d,1).
d(1) :- goal(d,c,e3) * del(goal(d,c,e3)) * e3.

e1 :- while_do(e1,1).
e1(1) :- action1.

e2 :- while_do(e2,1).
e2(1) :- e1.

e3 :- while_do(e3,1).
e3(1) :- e1.

action1:- write('action1'),nl.

while_do(Pred,N):- (FullPred =.. [Pred,N]) * execCTR(FullPred) * (N1 is N+1) * while_do(Pred,N1).
while_do(Pred,N):- true.
```

```
xsb -e "[load],init,ctr_comp('event_04')."

 | ?- execCTR(a),execCTR(b).
 action1

| ?- execCTR(c),execCTR(d).
 action1
```

Original program:

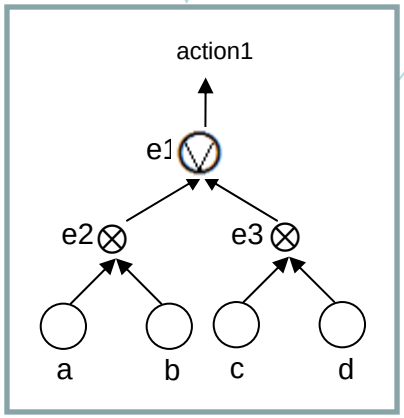
a([T1,T2]) * b([T3,T4]) -> e2([T1,T4])

c([T1,T2]) * d([T3,T4]) -> e3([T1,T4])

e2([T1,T2]) \/ e3([T1,T2]) -> e1([T1,T2])

Action triggers:

e1([T1,T2]) -> action1([T1,T2]).




```
a([T1,T2]) :- while_do(a,1,[T1,T2]).
a(1,[T1,T2]) :- ins(goal(b([_,_]),a([T1,T2]),e2([_,_]))).

b([T1,T2]) :- while_do(b,1,[T1,T2]).
b(1,[T3,T4]) :- goal(b([T3,T4]),a([T1,T2]),e2([_,_])) * del(goal(b([T3,T4]),a([T1,T2]),e2([_,_]))) * e2([T1,T4]).

c([T1,T2]) :- while_do(c,1,[T1,T2]).
c(1,[T1,T2]) :- ins(goal(d([_,_]),c([T1,T2]),e3([_,_]))).

d([T1,T2]) :- while_do(d,1,[T1,T2]).
d(1,[T3,T4]) :- goal(d([T3,T4]),c([T1,T2]),e3([_,_])) * del(goal(d([T3,T4]),c([T1,T2]),e3([_,_]))) * e3([T1,T4]).

e1([T1,T2]) :- while_do(e1,1,[T1,T2]).
e1(1,[T1,T2]) :- action1.

e2([T1,T2]) :- while_do(e2,1,[T1,T2]).
e2(1,[T1,T2]) :- e1([T1,T2]).

e3([T1,T2]) :- while_do(e3,1,[T1,T2]).
e3(1,[T1,T2]) :- e1([T1,T2]).

action1([T1,T2]):- write(action1([T1,T2])),nl.

while_do(Pred,N,L):- (FullPred =.. [Pred,N,L]) * execCTR(FullPred) * (N1 is N+1) * while_do(Pred,N1,L).
while_do(Pred,N,L):- true.
```

Disjunction with times

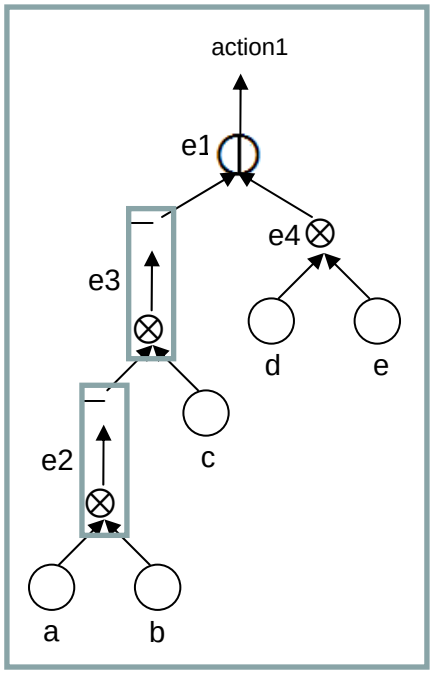
```
xsb -e "[load],init,ctr_comp('event_04_withTimes')."

 ?- execCTR(a([1,1])),execCTR(b([2,2])),
 action1([1,2])

 ?- execCTR(c([1,1])),execCTR(d([2,2])),
 action1([1,2])
```

Original program:

¯(a x b x c) | (d x e) -> e1

Action triggers:

e1 -> action1.



Note: isolation is propagated to all operations below

Isolation

```
a :- while_do(a,1).
a(1) :- ins(goal(b,a,iso(e2))).

b :- while_do(b,1).
b(1) :- goal(b,a,iso(e2)) * del(goal(b,a,iso(e2))) * e2.

c :- while_do(c,1).
c(1) :- goal(c,e2,iso(e3)) * del(goal(c,e2,iso(e3))) * e3.

d :- while_do(d,1).
d(1) :- ins(goal(e,d,e4)).
d(1) :- true.
d(2) :- goal(X,Y,iso(Z)), del(goal(X,Y,iso(Z))).

e :- while_do(e,1).
e(1) :- goal(e,d,e4) * del(goal(e,d,e4)) * e4.
e(1) :- true.
e(2) :- goal(X,Y,iso(Z)), del(goal(X,Y,iso(Z))).

e1 :- while_do(e1,1).
e1(1) :- action1.

e2 :- while_do(e2,1).
e2(1) :- ins(goal(c,e2,iso(e3))).

e3 :- while_do(e3,1).
e3(1) :- goal(e3,e4,e1) * del(goal(e3,e4,e1)) * e1.
e3(1) :- not(goal(e3,e4,e1)) * ins(goal(e4,e3,e1)).

e4 :- while_do(e4,1).
e4(1) :- goal(e4,e3,e1) * del(goal(e4,e3,e1)) * e1.
e4(1) :- not(goal(e4,e3,e1)) * ins(goal(e3,e4,e1)).

action1:- write('action1'),nl.

while_do(Pred,N):- (FullPred =.. [Pred,N]) * execCTR(FullPred) * (N1 is N+1) * while_do(Pred,N1).
while_do(Pred,N):- true.
```

```
xsb -e "[load],init,ctr_comp('event_05')."

?- execCTR(a),execCTR(b),execCTR(c),execCTR(d), execCTR(e).
 action1

?- execCTR(d), execCTR(e),execCTR(a),execCTR(b),execCTR(c).
 action1
```

Original program: ¯(a x b x c) | (d x e) -> e1

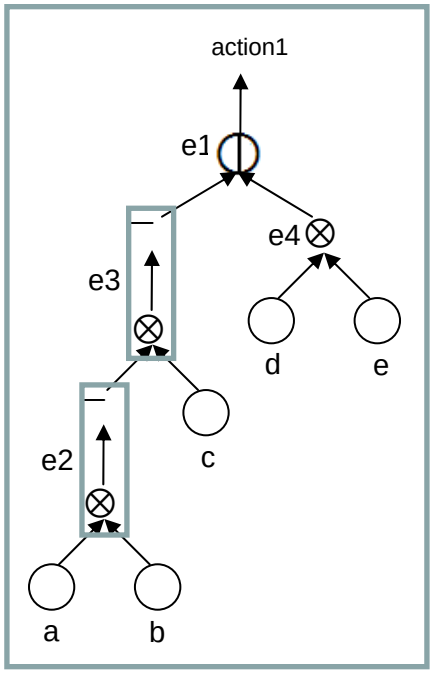
o(a([T1,T2]) * b([T3,T4]) * c([T5,T6]))-> e3([T1,T6])

d([T1,T2]) * e([T3,T4]) -> e4([T1,T4])

e3([T1,T2]) | e4([T1,T2]) -> e1([T1,T2])

Action triggers:

e1([T1,T2]) -> action1([T1,T2]).



Note: isolation is propagated to all operations below

```
a([T1,T2]) :- while_do(a,1,[T1,T2]).
a(1,[T1,T2]) :- ins(goal(b([_,_]),a([T1,T2]),iso(e2([_,_])))).

b([T1,T2]) :- while_do(b,1,[T1,T2]).
b(1,[T3,T4]) :- goal(b([T3,T4]),a([T1,T2]),iso(e2([_,_]))) * del(goal(b([T3,T4]),a([T1,T2]),iso(e2([_,_])))) * e2([T1,T4]).

c([T1,T2]) :- while_do(c,1,[T1,T2]).
c(1,[T3,T4]) :- goal(c([T3,T4]),e2([T1,T2]),iso(e3([_,_]))) * del(goal(c([T3,T4]),e2([T1,T2]),iso(e3([_,_])))) *
e3([T1,T4]).

d([T1,T2]) :- while_do(d,1,[T1,T2]).
d(1,[T1,T2]) :- ins(goal(e([_,_]),d([T1,T2]),e4([_,_]))).
d(1,[T1,T2]) :- true.
d(2,[T1,T2]) :- goal(X,Y,iso(Z)), del(goal(X,Y,iso(Z))).

e([T1,T2]) :- while_do(e,1,[T1,T2]).
e(1,[T3,T4]) :- goal(e([T3,T4]),d([T1,T2]),e4([_,_])) * del(goal(e([T3,T4]),d([T1,T2]),e4([_,_]))) * e4([T1,T4]).
e(1,[T1,T2]) :- true.
e(2,[T1,T2]) :- goal(X,Y,iso(Z)), del(goal(X,Y,iso(Z))).

e1([T1,T2]) :- while_do(e1,1,[T1,T2]).
e1(1,[T1,T2]) :- action1([T1,T2]).

e2([T1,T2]) :- while_do(e2,1,[T1,T2]).
e2(1,[T1,T2]) :- ins(goal(c([_,_]),e2([T1,T2]),iso(e3([_,_])))).

e3([T1,T2]) :- while_do(e3,1,[T1,T2]).
e3(1,[T3,T4]) :- goal(e3([T3,T4]),e4([T1,T2]),e1([_,_])) * del(goal(e3([T3,T4]),e4([T1,T2]),e1([_,_]))) * e1([T1,T4]).
e3(1,[T3,T4]) :- not(goal(e3([T3,T4]),e4([_,_]),e1([_,_]))) * ins(goal(e4([_,_]),e3([T3,T4]),e1([_,_]))).

e4([T1,T2]) :- while_do(e4,1,[T1,T2]).
e4(1,[T3,T4]) :- goal(e4([T3,T4]),e3([T1,T2]),e1([_,_])) * del(goal(e4([T3,T4]),e3([T1,T2]),e1([_,_]))) * e1([T1,T4]).
e4(1,[T3,T4]) :- not(goal(e4([T3,T4]),e3([_,_]),e1([_,_]))) * ins(goal(e3([_,_]),e4([T3,T4]),e1([_,_]))).

action1([T1,T2]):- write(action1([T1,T2])),nl.

while_do(Pred,N,L):- (FullPred =.. [Pred,N,L]) * execCTR(FullPred) * (N1 is N+1) * while_do(Pred,N1,L).
while_do(Pred,N,L):- true.
```

Isolation with times

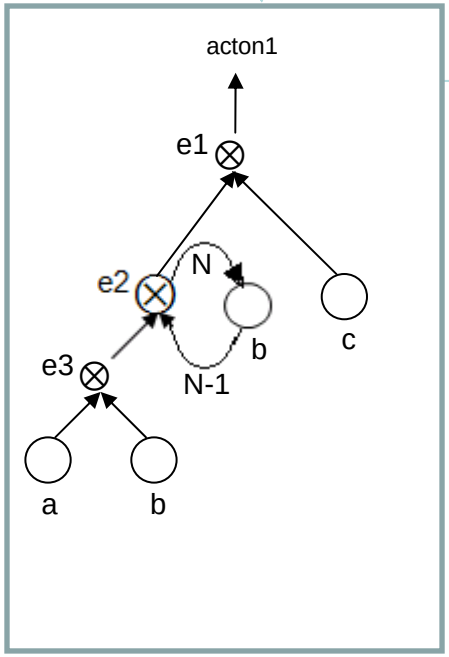
```
xsb -e "[load],init,ctr_comp('event_05_withTimes')."

?- execCTR(a([1,1])),execCTR(b([2,2])),execCTR(c([3,3])),execCTR(d([4,4])),execCTR(e([5,5])).
 action1([1,5])

?- execCTR(d([1,1])),execCTR(e([2,2])),execCTR(a([3,3])),execCTR(b([4,4])),execCTR(c([5,5])).
 action1([1,5])
```

Original program: a * b^10 * c -> e1

Informally, event a was fired, b was fired 10 times, followed by event c, triggering event e1 .

Note: b^N can also be represented as:

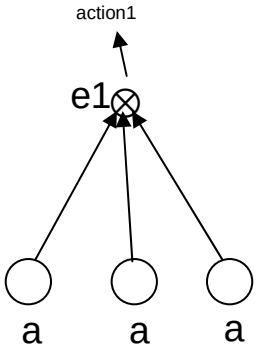
b * b * … * b (N times)

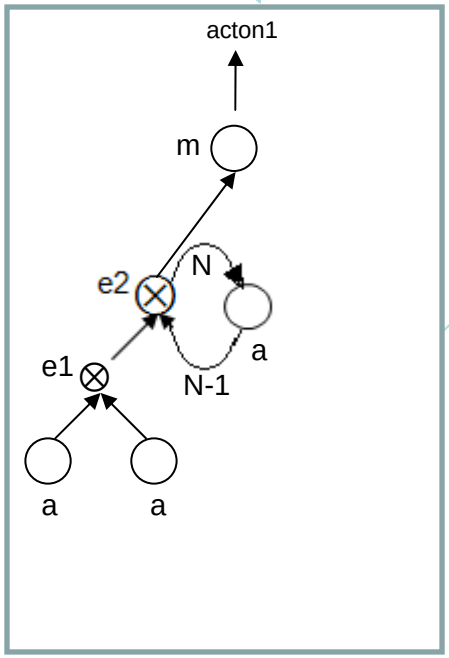


N-times

```
a :- while_do(a,1).
a(1) :- ins(goal(b,a,e3)).

b :- while_do(b,1).
b(1) :- goal(b,a,e3) * del(goal(b,a,e3)) * e3.
b(1) :- true.
b(2) :- goal(e3,e2,N) * del(goal(e3,e2,N)) * e2(N).

c :- while_do(c,1).
c(1) :- goal(e2,c,e1) * del(goal(e2,c,e1)) * e1.

e1 :- while_do(e1,1).
e1(1) :- action1.

e2(N) :- while_do(e2,1, N).
e2(1,N) :- N = 0 * ins(goal(e2,c,e1)).
e2(1,N) :- N > 0 * ins(goal(e3,e2,N-1)).

e3 :- while_do(e3,1).
e3(1) :-  N = 10, N1 is N-1, ins(goal(e3,e2,N1)).  % N is the constant 10

action1:- write('action1'),nl.

while_do(Pred,N):- (FullPred =.. [Pred,N]) * execCTR(FullPred) * (N1 is N+1) * while_do(Pred,N1).
while_do(Pred,N):- true.
```

```
xsb -e "[load],init,ctr_comp('event_06‘)."

?- execCTR(a) ,execCTR(b) ,execCTR(b) ,execCTR(b) ,execCTR(b) ,execCTR(b) ,execCTR(b) ,execCTR(b)
,execCTR(b) ,execCTR(b) ,execCTR(b) ,execCTR(c).
 action1
```





Original program:

a x a x a -> m

Action triggers:

m -> action1.

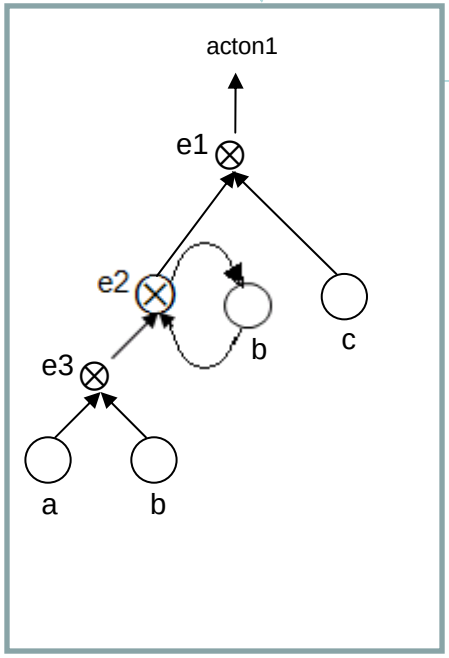
(N-times with time interval)

```
a :- while_do(a,1).
a(1, [T1,T2]) :- ins(goal(a([_,_]),a([T1,T2]),e1([_,_]))).

a(2,[T3,T4]) :- goal(a[T3,T4],a[T1,T2],e1([_,_])) * T2<T3 *
del(goal(a([T3,T4]),a([T1,T2]),e1)) * e1([T1,T4]).

a(3,[T3,T4]) :- goal(a([T3,T4]),e1([T1,T2]),e2([_,_])) * T2<T3 *
del(goal(a([T3,T4]),e1([T1,T2]),e1)) * e2([T1,T4]).

a(4,[T3,T4]) :- goal(a([T3,T4]),e2([T1,T2]),m([_,_])) * T4<T2 *
del(goal(a([T3,T4]),e2([T1,T2]),m(_,_))) * m([T1,T4]).

e1([T1,T2]) :- ins(goal(a([_,_]),e1([T1,T2]),e2([_,_]))).

e2([T1,T2]) :- ins(goal(a([_,_]),e2([T1,T2]),m([_,_]))).

e1 :- while_do(e1,1).
e1(1) :- action1.

while_do(Pred,N):- (FullPred =.. [Pred,N]) * execCTR(FullPred) * (N1 is N+1)
* while_do(Pred,N1).
while_do(Pred,N):- true.
```

```
xsb -e "[load],init,ctr_comp('event_01')."

 | ?- execCTR(a).
 yes
 | ?- execCTR(a).
 yes
 | ?- execCTR(a).
 action1
yes
```

Original program: a * b^* * c -> e1

Informally, event a was fired, b was fired at least once, followed by event c, triggering event e1 .



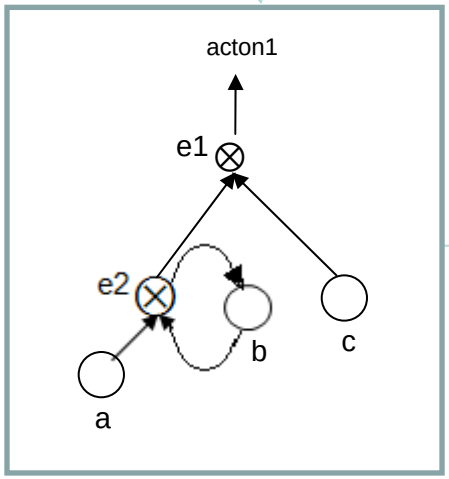

```
a :- while_do(a,1).
a(1) :- ins(goal(b,a,e3)).

b :- while_do(b,1).
b(1) :- goal(b,a,e3) * del(goal(b,a,e3)) * e3.
b(1) :- true.
b(2) :- goal(e2,c,e1) * del(goal(e2,c,e1)) * e2.

c :- while_do(c,1).
c(1) :- goal(e2,c,e1) * del(goal(e2,c,e1)) * e3.

e1 :- while_do(e1,1).
e1(1) :- action1.

e2 :- while_do(e2,1, N).
e2(1) :- ins(goal(e2,c,e1)).

e3 :- while_do(e3,1).
e3(1) :- ins(goal(e3,b,e2)).

action1:- write('action1'),nl.

while_do(Pred,N):- (FullPred =.. [Pred,N]) * execCTR(FullPred) * (N1 is N+1) *
while_do(Pred,N1).
while_do(Pred,N):- true.
```

*-times

```
xsb -e "[load],init,ctr_comp('event_06'_star)."

?- execCTR(a) ,execCTR(b) , execCTR(b) ,execCTR(c).
 action1
```

Original program: a * b^* * c -> e1

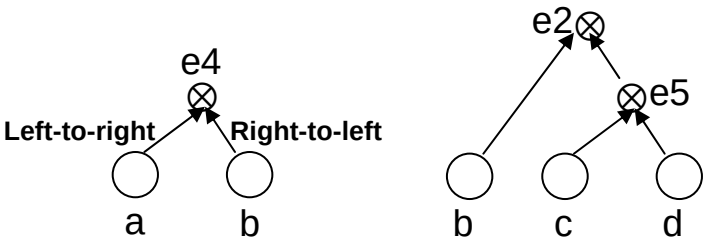
Informally, event a was fired, b was fired at least once, followed by event c, triggering event e1 .




```
a :- while_do(a,1).
a(1,[T1,T2]) :- ins(goal(b([_,_]),a([T1,T2]),e2([_,_]))).

b([T3,T4]) :- while_do(b([T3,T4]),1).

b(1,[T3,T4]) :- findall( goal(b([_,_]),a([T1,T2]),e3([_,_])),goal(b([_,_]),a([T1,T2]),e3([_,_])),L),
                while_do(member(goal(b([_,_]),a([T1,T2]),e3([_,_])),L) (
                        T2<T3 * del(goal(b([_,_]),a([T1,T2]),e3([_,_]))) * e2([T1,T4]) ,
                 ).

b(2,[T3,T4]) :-
   findall(goal(c([X,Y]),e2([T1,T2]),e1([Z,W])),goal(c([X,Y]),e2([T1,T2]),e1([Z,W])),L),
                while_do(member(goal(c([X,Y]),e2([T1,T2]),e1([Z,W])),L) (
                        del(goal(c([X,Y]),e2([T1,T2]),e1([Z,W]))) * e2([T1,T4])
                 ).

c :- while_do(c,1).
c(1, ([T3,T4])) :- goal(c([X,Y]),e2([T1,T2]), e1([Z,W])) * del(goal(c([X,Y]),e2([T1,T2]),
e1([Z,W]))) * e1([T1,T4]).

e1 :- while_do(e1,1).
e1(1, [T1,T2]) :- action1([T1,T2]).

e2 :- while_do(e2,1, N).
e2(2,[T1,T2]) :- ins(goal(c([_,_]),e2([T1,T2]), e1([_,_]))).

action1([T1,T2]):- write('action1'),nl.

while_do(Pred,N):- (FullPred =.. [Pred,N]) * execCTR(FullPred) * (N1 is N+1) *
while_do(Pred,N1).
while_do(Pred,N):- true.
```

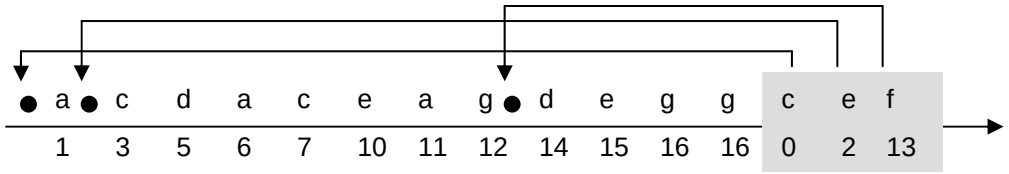
Fixed *-times

```
xsb -e "[load],init,ctr_comp('event_06‘_star)."

?- execCTR(a) ,execCTR(b) , execCTR(b) ,execCTR(c).
 action1
```


e4
Left-to-right
Right-to-left
a
b

e2
e5
b
c
d

Processing Out-of-Order Events



Received Order of Events